Python 程式設計　講義

串列、元組與物件參照

滄海《Python 程式設計初學指引》二版 · CH05

§ 1　名詞速查表

中文	English	一句話定義	例	易混
序列型別物件	sequence	元素有固定排位順序、可用序號存取的物件	串列、元組、字串	集合無序、不可索引
串列	list	**可變**的有序序列，用 `[]` 含括	`[400, 300, 200]`	與元組差在可不可變
元組	tuple	**不可變**的有序序列，用 `()` 含括	`(400, 300, 200)`	宣告後不能增刪改
可變序列	mutable sequence	元素數量與內容宣告後可改	串列	元組是不可變
不可變序列	immutable sequence	元素數量、內容、順序宣告後皆不可改	元組、字串	—
索引	index	元素的位置序號，從 `0` 起算，可用負數從尾算	`MyBook[0]`、`MyBook[-1]`	序號從 0 不是 1
切片	slicing	取某區間元素 `[起始:結束:梯級]`，**不含結束序號**	`L[1:3]`、`L[::2]`	含頭不含尾
梯級	step	切片時跳躍讀取的間隔，預設 1，可為負	`L[::2]`、`L[::-1]`	—
參照位置	reference	變數實際指向的資料所在記憶體位置（`id()` 可看）	`id(MyBook)`	變數存的是位址不是資料本身
別名	alias	多個變數名指向同一個物件	`b = a` 後 a、b 同物件	改一個另一個也變
淺複製	shallow copy	只複製第一層、巢狀子物件仍共享	`ListB[:]`、`copy.copy()`	子串列會連動
深複製	deep copy	連同所有層子物件都複製，完全獨立	`copy.deepcopy()`	需 `import copy`

§ 2 核心概念

核心概念

資料結構 (data structure) 是「有規則地存放一群資料」。Python 最常用四種：串列 (list)、元組 (tuple)、字典 (dict)、集合 (set)。本章是前兩者。

串列與元組都是**序列型別物件**：元素有排位順序，可用序號 (index) 取值。兩者用法幾乎一樣，差別只有一個關鍵字——**可不可變 (mutable)**。串列是可變序列，宣告後可隨時新增、刪除、修改元素；元組是不可變序列，宣告後元素的數量、內容、順序都鎖死。寫程式時的選擇準則：資料會變動用串列，固定不動（如座標、設定）用元組，元組更省記憶體也更安全。

第二個關鍵概念貫穿整章：**Python 的變數是「名牌」不是「盒子」**。`MyPrice = [400, 300]` 不是把資料裝進名為 MyPrice 的盒子，而是先在記憶體建一個 list 物件，再把 `MyPrice` 這張名牌貼上去。`=` 是「貼名牌（指派參照位置）」，不是「拷貝資料」。理解這點，別名、淺複製、深複製就都通了——這也是本章最後、份量最重的考點。

§ 3 主要內容

3.1 串列：宣告與型別轉換

串列用中括弧含括、逗號分隔，元素可混型別、可巢狀（元素本身又是串列／元組／字典）：

```
MyList = []                                     # 空串列
MyPrice = [400, 300, 200, 100]                  # 數值
MyBook = ['Python', 'Java', 'VB', 'C++']        # 字串
MyBookCost = ['Python', 400, 'Java', 300]       # 混型別
MyMixList = ['Python', 400, ['van Rossum', 1989]]   # 巢狀
```

`list(可迭代物件)` 可把元組、字典（取 key）、集合、字串拆成串列：

```
list((100, 200, 300))           # [100, 200, 300]      元組→串列
list({'A':400, 'B':500})        # ['A', 'B']           字典→取 key
list({700, 800, 900})           # [800, 900, 700]      集合（無序）
list('Anaconda')                # ['A','n','a','c',...] 字串逐字元
```

3.2 讀取元素：索引與切片

序號從 `0` 起算，`-1` 指最後一個。巢狀用連續中括弧逐層取：

```
MyMixList1 = ['Python', 400, ['van Rossum', 1989], ('Google', 'Dropbox')]
MyMixList1[2]        # ['van Rossum', 1989]  取出內層串列
MyMixList1[2][0]     # 'van Rossum'          再取內層第 0 個
MyMixList1[3][0]     # 'Google'              內層元組第 0 個
```

切片 `串列[起始:結束:梯級]` 取一段，**含起始、不含結束**；梯級預設 1：

```
MyBookCost = ['Python', 400, 'Java', 300, 'VB', 200, 'C++', 100]
MyBookCost[::2]      # ['Python','Java','VB','C++']  每隔一個（書名）
MyBookCost[1::2]     # [400, 300, 200, 100]          從序號1每隔一個（價格）
MyBookCost[2:5]      # ['Java', 300, 'VB']           序號 2~4（不含 5）
```

省略規則：起始省略＝從頭、結束省略＝到尾、都省略＝全部；梯級可負（`[::-1]` 反轉）。

3.3 修改、增加、刪除元素（只有串列能做）

```
MyPrice = [400, 300, 200, 100]
MyPrice[1] = 350            # 改單一元素 → [400, 350, 200, 100]
MyPrice[::2] = [450, 250]   # 改一個區間 → [450, 350, 250, 100]

MyBook.append('R')          # 加在最末端
MyBook.insert(1, 'Julia')   # 插到序號 1，後面元素往後移

MyBook.pop()                # 移除並「回傳」最後一個
MyBook.pop(1)               # 移除並回傳序號 1 的元素
MyBook.remove('Java')       # 刪掉第一個值為 'Java' 的元素（回傳 None！）
del MyBook[2]               # del 語句刪序號 2
del MyBook[1::2]            # del 刪一個區間
del MyBook                  # 連整個變數一起刪除（之後再用會 NameError）
```

關鍵區別

`pop()` 會**回傳**被移除的元素（可接住來用）；`remove()` 與 `del` **不回傳**（`remove()` 回傳 `None`）。所以 `print(MyBook.remove('Java'))` 印出的是 `None`，不是被刪的元素。

3.4 元組：宣告與單元素逗號陷阱

元組用小括弧，用法與串列相同，但宣告後不可變：

```
Tup = ()                                    # 空元組
TupPrice = (400, 300, 200, 100)
TupBook = ('Python', 'Java', 'VB', 'C++')
TupMix = ('Python', 400, ['van Rossum', 1989])   # 可巢狀
```

單一元素元組必加逗號

`(10,)` 才是元組；`(10)` 只是「被括弧包住的整數 10」。括弧在這裡會被當成運算優先符號，要靠**結尾逗號**才認得出是元組。

```
TupHasComma = (10,)     # type → <class 'tuple'>
TupNoComma  = (10)      # type → <class 'int'>   (不是元組！)
```

`tuple(可迭代物件)` 反向把串列等轉成元組（`tuple([100,200,300])` → `(100, 200, 300)`）。

3.5 進階操作：統計、排序、搜尋、結合

類別	串列做法	重點
統計	`len()` `max()` `min()` `sum()`	`len` 可用於混型別；`max/min` 限純數值或純字串；`sum` 限數值
排序	`L.sort()` `L.reverse()` `sorted(L)`	`sort/reverse` **原地改、回傳 None**；`sorted()` **回傳新串列、不動原串列**
逆向排序	`L.sort(reverse=True)`	`sorted(L, reverse=True)` 同理但不動原串列
搜尋計次	`L.index(x)` `L.count(x)`	`index` 回傳**第一個**出現的序號，找不到 **ValueError**；`count` 回傳出現次數
結合	`L1.extend(L2)`、`L1 + L2`、`L * n`	`+` 與 `*` 產生新串列；`extend` 原地改 L1
串成字串	`'分隔'.join(L)`	由字串物件提供，元素須為字串

```
MyBook.sort()                       # 原地排序，MyBook 本身變了
MySorted = sorted(MyBook)           # MySorted 是新串列，MyBook 不變
' || '.join(['Python','Java'])      # 'Python || Java'
[400, 300] * 2                      # [400, 300, 400, 300]
```

3.6 招牌：物件參照、淺複製與深複製

這是本章份量最重、最容易考的部分。回到核心概念——**變數是名牌**。

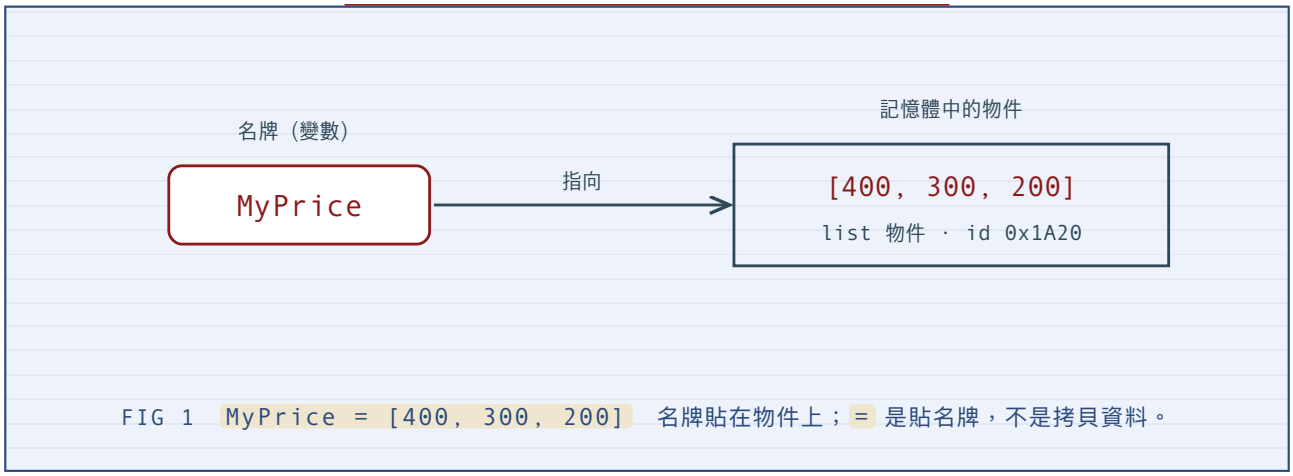

FIG 1 `MyPrice = [400, 300, 200]` 名牌貼在物件上；`=` 是貼名牌，不是拷貝資料。

別名 (alias)：`=` 只是再貼一張名牌到**同一個**物件。兩張名牌共享一份資料，改任一個，另一個也跟著變（`is` 為 `True`、`id` 相同）。要得到「內容相同但獨立」的新物件，得用切片 `[:]` 之類的方式複製。

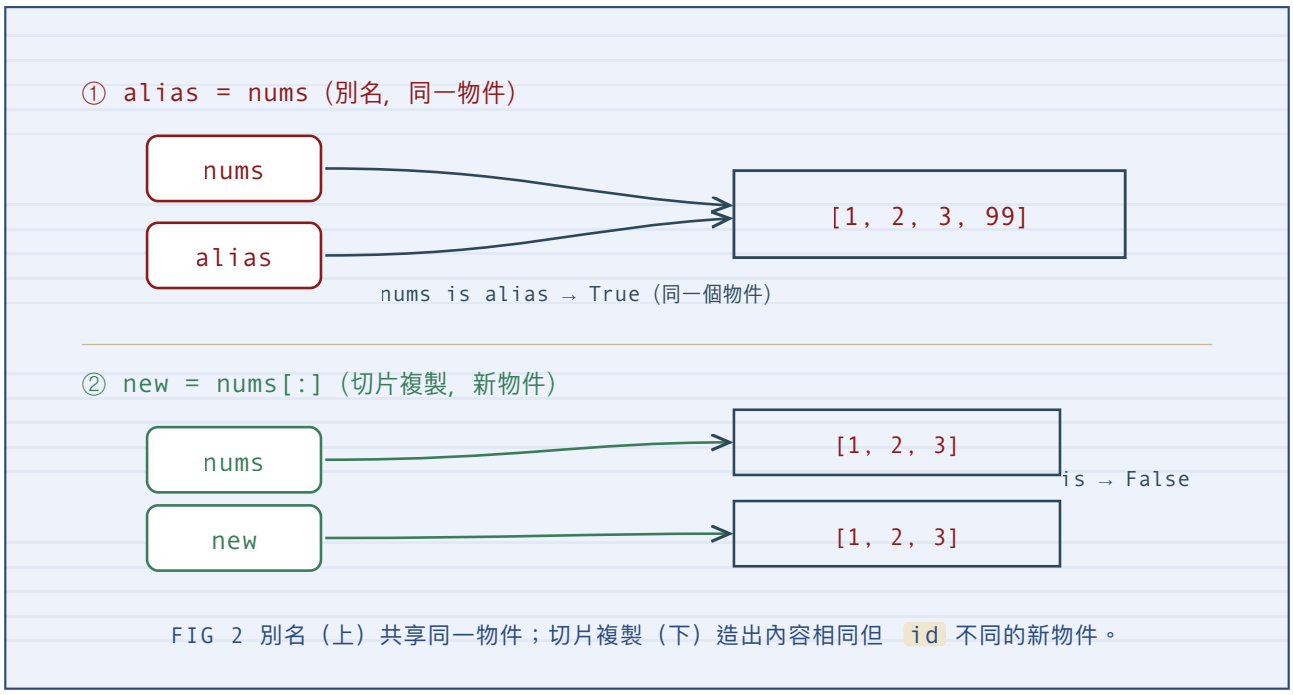

FIG 2 別名（上）共享同一物件；切片複製（下）造出內容相同但 `id` 不同的新物件。

淺複製 vs 深複製：當串列裡又包著子串列（巢狀），差別才會浮現。`ListB[:]` 是**淺複製**——只複製最外層，內層子串列仍與原串列**共享同一個**。所以改子串列的元素，兩邊都變。

`copy.deepcopy()` 是**深複製**——連內層都另外複製一份，徹底獨立。

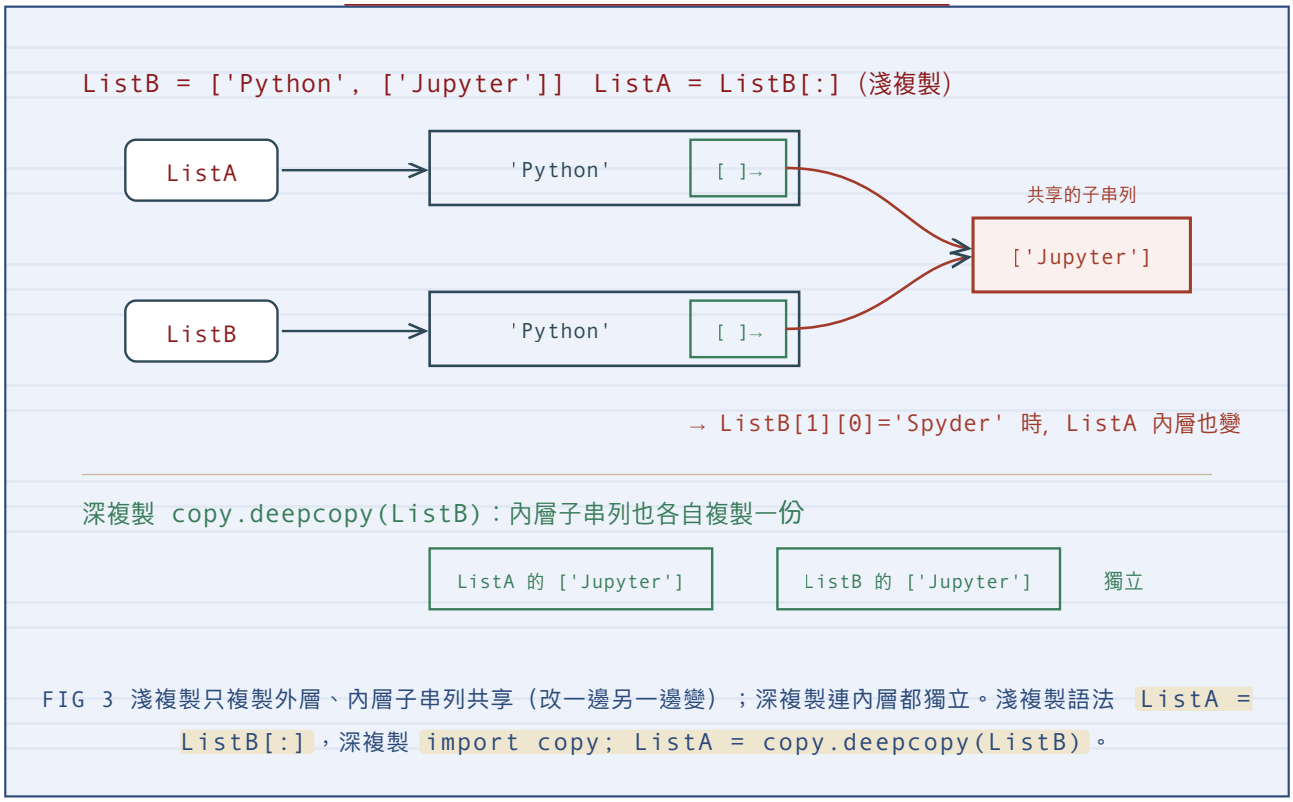

FIG 3 淺複製只複製外層、內層子串列共享（改一邊另一邊變）；深複製連內層都獨立。淺複製語法 `ListA = ListB[:]`，深複製 `import copy; ListA = copy.deepcopy(ListB)`。

3.7 元組的「不可變」有層次

元組本身不可改（不能 `Tup[1] = x`、沒有 `sort()`／`append()`），但若元組裡裝著**可變元素**（串列、字典），那個可變元素的內容仍可改——因為改的是子物件，不是元組的參照：

```
TupMix = ('Python', 400, ['VB', 'C++'], {'Julia': 3})
TupMix[2].append('.Net')   # OK！改的是內層串列
TupMix[3]['Julia'] = 5     # OK！改的是內層字典
TupMix[1] = 500            # 報錯 TypeError：直接改元組元素不行
```

元組沒有 `sort()`／`reverse()`，但可用 `sorted()`（回傳排序後的「串列」，原元組不動）；`index()`／`count()`／`len()`／`max()`／`min()`／`sum()` 元組都有。

§ 4 語法與方法速查

```
# 建立
L = [元素, ...]          T = (元素, ...)        T1 = (單元素,)    # 單元素元組要逗號
list(可迭代物件)          tuple(可迭代物件)

# 取值
L[i]   L[-1]             L[起始:結束:梯級]      L[2][0]           # 含頭不含尾、巢狀逐層

# 串列才有的「會改變自己」的操作（多半回傳 None）
L[i] = x                 L.append(x)   L.insert(i, x)
L.pop()→回傳  L.pop(i)→回傳   L.remove(x)→None   del L[i]   del L[區間]   del L
L.sort()  L.sort(reverse=True)   L.reverse()   L.extend(L2)

# 不改原物件、回傳新結果
sorted(L)   sorted(L, reverse=True)   L1 + L2   L * n   '分隔'.join(L)

# 查詢（串列與元組皆有）
len(L)  max(L)  min(L)  sum(L)   L.index(x)   L.count(x)

# 複製三層次
A = B                    # 別名：同一物件，連動
A = B[:]                 # 淺複製：外層獨立、內層共享
import copy; A = copy.deepcopy(B)   # 深複製：完全獨立
```

寫法	結果	原物件
`L.sort()`	`None`（原地排序）	**被改**
`sorted(L)`	新的已排序串列	不變
`L.remove(x)`	`None`	**被改**
`L.pop(i)`	被移除的元素	**被改**
`B[:]`	新串列（淺）	不變

§ 5 常見錯誤

常見錯誤

- `remove()` ／ `sort()` ／ `reverse()` **回傳** `None`：`x = L.sort()` 會讓 `x` 變 `None`；要新串列用 `sorted(L)`。
- **切片含頭不含尾**：`L[1:3]` 取序號 1、2，不含 3。
- **單一元素元組漏逗號**：`(10)` 是整數，`(10,)` 才是元組。
- **集合不支援索引**：串列裡若放集合，`L[i][0]` 會 `TypeError: 'set' object is not subscriptable`。
- `index()` **找不到會丟** `ValueError`：先用 `in` 判斷存在再 `index`，或用 `count` 確認次數。
- **元組不可變的兩種報錯**：`Tup[1]=x` → `TypeError: 'tuple' object does not support item assignment`；`Tup.sort()` → `AttributeError: 'tuple' object has no attribute 'sort'`。
- `=` **不是複製**：`b = a` 後兩者同一物件，改 `b` 會動到 `a`。要獨立用 `a[:]`（淺）或 `copy.deepcopy`（深）。
- **淺複製的巢狀陷阱**：`a = b[:]` 後改 `b` 的子串列元素，`a` 也跟著變（內層共享）。
- **可變物件當函式預設參數**：`def f(x, lst=[])` 的 `lst` 跨呼叫共用同一個串列，會累積——預設值用 `None`，函式內再建新串列。

§ 6 練習題

練習 1（一般題）：切片預測

> **引導步驟**
>
> 1. 標出每個元素的正、負序號。
> 2. `[1:5:2]` 從序號 1 開始、不含 5、每隔 2 個取。
> 3. `[::-1]` 梯級 -1 代表什麼方向？

```
L = [10, 20, 30, 40, 50, 60]
print(L[1:5:2])
print(L[-2:])
print(L[::-1])
```

解答

`[20, 40]`（序號 1、3）／`[50, 60]`（倒數兩個）／`[60, 50, 40, 30, 20, 10]`（反轉）。

> **易錯**
>
> · 把 `[1:5:2]` 當成取到序號 5；切片不含結束序號。

練習 2（一般題）：增刪改方法

引導步驟

1. `append` ／ `insert` 之後串列長什麼樣？
2. `pop(1)` 回傳什麼、串列剩什麼？
3. `print(L.remove('VB'))` 印出的是什麼？

```
L = ['Python', 'Java', 'VB']
L.append('C++')
L.insert(1, 'Julia')
print(L.pop(1))
print(L.remove('VB'))
print(L)
```

解答

`append` 後 `['Python','Java','VB','C++']`；`insert(1,'Julia')` 後 `['Python','Julia','Java','VB','C++']`。`pop(1)` 回傳並印出 `'Julia'`。`print(L.remove('VB'))` 印出 `None`（remove 不回傳被刪元素）。最後 `L` 為 `['Python','Java','C++']`。

易錯

· 以為 `print(L.remove('VB'))` 會印出 `'VB'`；其實印 `None`。

練習 3（重要題）：淺複製 vs 深複製

引導步驟

1. 畫出 ListB 的外層與內層子串列兩個物件。
2. `ListB[:]` 淺複製：外層另建、內層子串列是否共享？
3. 改 `ListB[1][0]`，ListA 會不會變？深複製呢？

```
import copy
ListB = ['Python', ['Jupyter']]
ListA = ListB[:]              # 淺複製
ListC = copy.deepcopy(ListB)  # 深複製
ListB[1][0] = 'Spyder'
print(ListA)
print(ListC)
```

解答

`ListA` → `['Python', ['Spyder']]`（淺複製內層共享，跟著變）；`ListC` → `['Python', ['Jupyter']]`（深複製內層獨立，不變）。關鍵：淺複製只複製最外層，內層子串列仍指向同一物件。

易錯

· 以為 `[:]` 完全獨立；對巢狀只獨立了外層。要全獨立用 `copy.deepcopy`。

練習 4（重要題）：元組的不可變層次

引導步驟

1. `TupMix[2]` 是什麼型別？它可不可變？
2. `TupMix[2].append(...)` 改的是元組還是子物件？
3. `TupMix[1] = 500` 改的是元組本身的元素，會怎樣？

```
TupMix = ('Python', 400, ['VB', 'C++'])
TupMix[2].append('.Net')
print(TupMix)
TupMix[1] = 500
```

解答

`TupMix[2].append('.Net')` 成功，印出 `('Python', 400, ['VB', 'C++', '.Net'])`——改的是內層**串列**（可變），不是元組本身。`TupMix[1] = 500` 報錯 `TypeError: 'tuple' object does not support item assignment`——直接改元組元素不允許。

易錯

· 以為元組裡的東西全都不能動；其實裡面的可變子物件仍可改其內容。

練習 5（一般題）：sort 與 sorted

引導步驟

1. `sorted(L)` 回傳什麼、`L` 本身變了嗎？
2. `L.sort()` 回傳什麼、`L` 本身變了嗎？
3. 接住 `L.sort()` 的回傳值會是什麼？

```
L = [3, 1, 2]
a = sorted(L)
print(a, L)
b = L.sort()
print(b, L)
```

解答

`a = [1, 2, 3]`、`L` 仍是 `[3, 1, 2]`（sorted 不動原串列）。`b = None`、`L` 變成 `[1, 2, 3]`（sort 原地改、回傳 None）。

易錯

- 以為 `b = L.sort()` 會接到排序後的串列；其實接到 `None`。

§ 7 自我檢核

- [ ] 能說出串列與元組的唯一關鍵差異（可變 vs 不可變）與各自的括弧。
- [ ] 能正確切片：含頭不含尾、梯級、負索引、`[::-1]` 反轉。

- [ ] 能分辨會改原物件（`sort/reverse/append/remove/pop`）與回傳新結果（`sorted/+/*`）的操作。
- [ ] 記得 `remove()/sort()/reverse()` 回傳 `None`，`pop()` 回傳被移除元素。
- [ ] 能解釋「變數是名牌、`=` 是貼名牌不是拷貝」。
- [ ] 能畫出別名、淺複製、深複製三種情況的物件參照圖。
- [ ] 能預測淺複製對巢狀子串列的連動、深複製的獨立。
- [ ] 能說明元組「不可變」的層次：本身不可改，但內含的可變子物件可改。
- [ ] 知道單一元素元組要加逗號、集合不可索引、`index()` 找不到會 `ValueError`。